STAT 4410/8416 Homework 3

ISLAM MD TAHIDUL

Due on October 30

1. Text Data analysis: Download `lincoln-last-speech.txt` from Canvas which contains Lincoln's last public address. Now answer the following questions and include your codes.

a. Read the text and store the text in `lAddress`. Show the first 70 characters from the first element of the text.

```
lAddress <- readLines("lincoln-last-speech.txt", encoding = "UTF-8")
cat(substr(lAddress[1], 1, 70))
```

```
## We meet this evening, not in sorrow, but in gladness of heart. The eva
```

b. Now we are interested in the words used in his speech. Extract all the words from `lAddress`, convert all of them to lower case and store the result in `vWord`. Display first few words.

```
# Combine all lines into one long text
full_text <- paste(lAddress, collapse = " ")
# Tokenize the text into words
words <- unlist((strsplit(full_text, "\\s+")))
# Convert all words to lowercase
vWord <- tolower(words)
# Display the first few words
head(vWord)
```

```
## [1] "we"       "meet"     "this"     "evening," "not"      "in"
```

c. The words like `am`, `is`, `my` or `through` are not much of our interest and these types of words are called stop-words. The package `tm` has a function called `stopwords()`. Get all the English stop words and store them in `sWord`. Display few stop words in your report.

```
#install.packages("tm")
library(tm)
# Get English stop words
sWord <- stopwords("en")
# Display a few stop words
head(sWord)
```

```
## [1] "i"      "me"     "my"     "myself" "we"     "our"
```

d. Remove all the `sWord` from `vWord` and store the result in `cleanWord`. Display first few clean words.

```
# library(tm)
# Remove stop words from vWord
cleanWord <- vWord[!vWord %in% sWord]
# Display the first few clean words
head(cleanWord)
```

```
## [1] "meet"       "evening,"   "sorrow,"    "gladness"   "heart."
## [6] "evacuation"
```

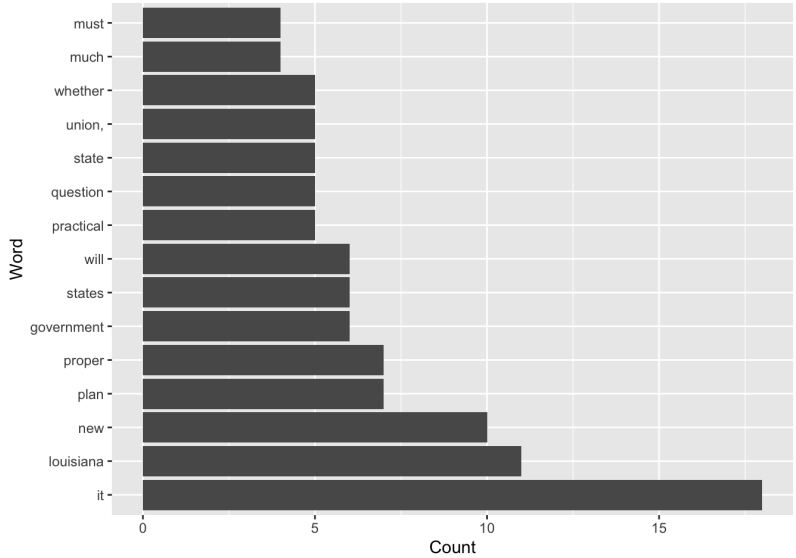
e. `cleanWord` contains all the cleaned words used in Lincoln's address. We would like to see which words are more frequently used. Find 15 most frequently used clean words and store the result in `fWord` . Display first 5 words from `fWord` along with their frequencies.

```
# Combine "it." and "it," as one word "it"
cleanWord[grepl("^(it\\.|it,)$", cleanWord)] <- "it"
# Create a table of word frequencies
word_freq <- table(cleanWord)
# Sort the table in descending order
sorted_word_freq <- sort(word_freq, decreasing = TRUE)
# Get the top 15 most frequently used words
fWord <- head(sorted_word_freq, 15)
# Display the first 5 words from fWord along with their frequencies
head(fWord,5)
```

```
## cleanWord
##        it louisiana       new      plan    proper
##        18        11        10         7         7
```

f. Construct a bar chart showing the count of each words for the 15 most frequently used words. Add a layer `+coord_flip()` with your plot.

```
library(ggplot2)
# Convert fWord to a data frame for plotting
word_15_most <- as.data.frame(fWord)
# Rename the columns for clarity
colnames(word_15_most) <- c("Word", "Count")
# Create a bar chart
plot <- ggplot(word_15_most, aes(x = Word, y = Count)) +
  geom_bar(stat = "identity") +
  coord_flip()
# Display the bar chart
print(plot)
```



g) What is the reason for adding a layer `+coord_flip()` with the plot in question (f). Explain what would happen if we would not have done that.

This `+coord_flip()` used in data visualization to change the orientation of the plot from vertical bars to horizontal bars, or vice versa. The primary reason for doing this is to improve the readability and presentation of the data, especially when we have a large number of bars or long labels.

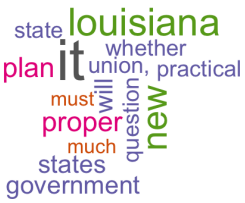
It can be problematic if we have a large number of words or long word labels.

h. The plot in question (f) uses bar plot to display the data. Can you think of another plot that delivers the same information but looks much simpler? Demonstrate your answer by generating such a plot.

```
#install.packages("wordcloud")
library(tm)
library(wordcloud)

# Create a data frame with word frequencies
wordcloud_data <- data.frame(word = names(fWord), freq = as.numeric(fWord))

# Generate a word cloud
wordcloud(words = wordcloud_data$word, freq = wordcloud_data$freq, scale=c(3, 0.5), c
olors=brewer.pal(8, "Dark2"))
```



i) In the question (c), you removed words that are called `stop-words` . Now please answer the following: a) Count the total stop words from `lAddress` and store it in `stopWordsCount`

```
# Count the stop words in the text
stopWordsCount <- sum(vWord %in% sWord)
stopWordsCount
```

```
## [1] 909
```

```
    b)  Count the total words (including stop-words) from `lAddress` and store it in
`lAddressCount`
```

```
# Tokenize the text into words
words <- unlist(strsplit(lAddress, "\\s+"))
# Count the total words (including stop words) in the text
lAddressCount <- length(words)
lAddressCount
```

```
## [1] 1815
```

```
    c)  Divide `stopWordsCount` by `lAddressCount` and report the percentage
```

```
# Calculate the percentage of stop words
percentage_stop_words <- round((stopWordsCount / lAddressCount) * 100,2)
cat(percentage_stop_words,"%")
```

```
## 50.08 %
```

```
    d)  Explain in your own words what does the percentage indicate in this context?
```

The percentage tells us that roughly half of the words in the text are non-substantive words.So, if we want to analysis text data then we should remove stop word to get meaningful insights.

2. Regular Expressions: Write a regular expression to match patterns in the following strings. Demonstrate that your regular expression indeed matched that pattern by including codes and results. Carefully review how the first problem is solved for you.

aa) We have a vector `vText` as follows. Write a regular expression that matches `g, og, go or ogo` in `vText` and replace the matches with '.'.

```
vText <- c('google','logo','dig', 'blog', 'boogie' )
```

Answer:

```
pattern <- 'o?go?'
gsub(pattern, '.', vText)
```

```
## [1] "..le"  "l."    "di."   "bl."   "bo.ie"
```

a. We have a vector `VNumbers` as follows. Write a regular expression that extracts only binary values (0,1) from `VNumbers` .

```
VNumbers = c('01011123AEX','1010183DIS','1A02L01A13', 'A2NE3000111')
# Create a vector of binary values using gsub
binary_values <- gsub("[^01]", "", VNumbers)
binary_values
```

```
## [1] "010111" "10101"  "10011"  "000111"
```

b. Replace only the 5 or 6 digit numbers with the word "found" in the following vector. Please make sure that 3, 4, or 7 digit numbers do not get changed.

```
vPhone = c('874','6783','345345', '32120', '468349', '8149674' )
replace_5or6 <-gsub("\\b\\d{5,6}\\b", "found", vPhone)
replace_5or6
```

```
## [1] "874"    "6783"   "found"  "found"  "found"  "8149674"
```

c. Replace all the characters that are not among the 26 English characters or a space. Please replace with an empty spring.

```
myText = "#y%o$u @g!o*t t9h(e) so#lu!tio$n c%or_r+e%ct"
replace_non_chr <- gsub("[^a-zA-Z ]","",myText)
replace_non_chr
```

```
## [1] "you got the solution correct"
```

d. Extract all the Proper nouns from the `myText` using Regular expression.

```
#problem for For
library(stringr)
myText = "welcome to University of Nebraska Omaha. For Math department classes please
go to DSC"
proper_nouns <- unlist(str_extract_all(myText, "\\b[A-Z][a-z]*\\b"))
proper_nouns
```

```
## [1] "University" "Nebraska"   "Omaha"      "For"        "Math"
```

e. Extract all the three numbers embedded in the following text.

```
#library(stringr)
bigText = 'There are four 20@20 numbers hid989den in the 500 texts'
extracted_numbers <- gsub("@", "", bigText)
extracted_numbers <- unlist(str_extract_all(extracted_numbers, "\\d+"))
#extracted_numbers <- str_extract_all(gsub("@", "", bigText), "\\d+")
extracted_numbers
```

```
## [1] "2020" "989"  "500"
```

f. Extract all the words that have a upper case letter in the start and end,convert all the words into lowercase.

```
myText = 'ThE SalrieS are ReporteD (in millions) for every CompanY.'
extract_uppeer_lower <- print(str_to_lower(unlist(str_extract_all(myText,"[A-Z][a-z]+
[A-Z]"))))
```

```
## [1] "the"      "salries"  "reported" "company"
```

g. Extract the texts in between _ and dot(.) in the following vector. Your output should be ‘bill’, ‘pay’, ‘fine-book’.

```
#..................
myText = c("H_bill.xls", "Big_H_pay.xls", "Use_case_fine-book.pdf")
extracted_texts <- str_extract(myText, "(?<=_)[^.]+")
# Remove any leading characters before "_"
extracted_texts <- sub(".*_", "", extracted_texts)

# Display the extracted texts
extracted_texts
```

```
## [1] "bill"     "pay"      "fine-book"
```

h. Extract the numbers (return only integers) that are followed by the units ‘ml’ or ‘lb’ in the following text.

```
myText = 'Received 10 apples with 200ml water at 8pm with 15 lb meat and 2lb salt'
unit_match <- as.integer(gsub("[a-z]*","",unlist(str_extract_all(myText,"[0-9]+\\s?[m
l|lb]+"))))
unit_match
```

```
## [1] 200  15   2
```

i. Extract only the word in between pair of symbols `$` . Count number of words you have found between pairs of dollar sign `$` .

```
myText = 'Math symbols are $written$ in $between$ dollar $signs$'
extract_dollar <- gsub("\\$","",unlist(str_extract_all(myText,"\\$([^\\$]+)\\$")))
extract_dollar
```

```
## [1] "written" "between" "signs"
```

```
cat("Number of words:",length(extract_dollar))
```

```
## Number of words: 3
```

j. Extract all the valid equations in the following text.

```
myText = 'equation1: 21 - 12 = 9, equation2 is: 2*3=6, do not extract 2w3=6'
valid_eq <-unlist(str_extract_all(myText, "\\b[0-9]*?\\s*[+\\-*\\/]+?\\s*[0-9]*?\\s*
[=]\\s*[0-9]*\\b"))
valid_eq
```

```
## [1] "21 - 12 = 9" "2*3=6"
```

k. Extract all the letters of the following sentence and check if it contains all 26 letters in the alphabet. If not, produce code that will return the total number of unique letters that are included and show the letters that are missing.

```
myText = 'there are five wizard boxing matches to be judged'
# Extract all letters from the sentence and convert to lowercase
input_letters <- tolower(gsub("[^a-z]", "", myText))
input_letters <-strsplit(input_letters, "")[[1]]
input_letters<-sort(unique(input_letters))
# Create a vector of all 26 lowercase alphabet letters
all_letters <- letters[1:26]
# Check if it contains all 26 letters
check_uniq <- all_letters %in% input_letters
if (!any(!check_uniq)) { # !any to check any false
  cat("The sentence contains all 26 letters of the alphabet.\n")
} else {
  missing_letters <- setdiff(unique(all_letters), unique(input_letters))
  cat("Total number of unique letters:",paste(length(missing_letters)))
  cat("\n")
  cat("Unique letters are:",missing_letters)
}
```

```
## Total number of unique letters: 5
## Unique letters are: k l p q y
```

l. Extract the valid web link from the text

```
myText = '<body> the valid site is http://www.bbc.com not http://.com'
valid_web_link <- unlist(str_extract_all(myText,"[a-z://]+[a-z]{3}[.][a-z]+[.][a-z]
{3}"))
valid_web_link
```

```
## [1] "http://www.bbc.com"
```

3. Extracting data from the web: Our plan is to extract data from web sources. This includes email addresses, phone numbers or other useful data. The function `readLines()` is very useful for this purpose.

a. Read all the text in https://www.unomaha.edu/college-of-arts-and-sciences/mathematics/about-us/directory/index.php (https://www.unomaha.edu/college-of-arts-and-sciences/mathematics/about-us/directory/index.php) and store your texts in `myText` . Show first few rows of `myText` and examine the structure of the data.

```
# URL of the web page
url <- "https://www.unomaha.edu/college-of-arts-and-sciences/mathematics/about-us/dir
ectory/index.php"
# Read the web page into a variable
myText <- readLines(url, warn = FALSE)
# Show the first few lines of myText
head(myText)
```

```
## [1] "<!DOCTYPE html>"
## [2] "<html xmlns=\"http://www.w3.org/1999/xhtml\" lang=\"en\">"
## [3] "\t<head>"
## [4] "\t\t<meta charset=\"utf-8\"/>"
## [5] "\t\t<meta content=\"Math Department Directory\" name=\"description\"/>"
## [6] "\t\t<meta content=\"width=device-width,initial-scale=1\" name=\"viewport\"/>"
```

b. Write a regular expression that would extract all the http web links addresses from `myText`. Include your codes and display the results that show only the `http` or `https` web link addresses and nothing else.

```
# Extract HTTP and HTTPS web link addresses
web_links <- regmatches(myText, gregexpr("https?://[^\t\n\r ()]+", myText))
# Remove any unwanted characters or whitespace
web_links <- gsub("[\t\n\r ()]", "", unlist(web_links))
# Display the extracted web link addresses
head(web_links)
```

```
## [1] "http://www.w3.org/1999/xhtml\""
## [2] "https://www.unomaha.edu/college-of-arts-and-sciences/mathematics/about-us/dir
ectory/index.php\""
## [3] "https://www.unomaha.edu/_files/uno-template/current/css/uno-template.css?dt=1
698264480\""
## [4] "http://www.w3.org/2000/svg\""
## [5] "http://www.w3.org/2000/svg\""
## [6] "http://www.w3.org/1999/xlink\">"
```

c. Now write a regular expression that would extract all the emails from `myText`. Include your codes and display the results that show only the email addresses and nothing else.

```
# Extract email addresses
email_addresses <- unique(unlist(regmatches(myText, gregexpr("\\b[A-Za-z0-9._+-]+@[A-
Za-z0-9.-]+\\.[A-Za-z]{2,}\\b", myText))))
#print email addresses
email_addresses
```

```
##  [1] "mbaccouch@unomaha.edu"        "rbrusky@unomaha.edu"
##  [3] "xycheng@unomaha.edu"          "jeffreydepue@unomaha.edu"
##  [5] "elder@unomaha.edu"            "sfrom@unomaha.edu"
##  [7] "keithgallagher@unomaha.edu"   "dholley@unomaha.edu"
##  [9] "yinghu@unomaha.edu"           "ninfante@unomaha.edu"
## [11] "christopherjames@unomaha.edu" "nkass@unomaha.edu"
## [13] "blove@unomaha.edu"            "mmajumder@unomaha.edu"
## [15] "vmatache@unomaha.edu"         "michaelmatthews@unomaha.edu"
## [17] "lmcfee@unomaha.edu"           "kenzimedeiros@unomaha.edu"
## [19] "lpalayangoda@unomaha.edu"     "lindarau@unomaha.edu"
## [21] "jrech@unomaha.edu"            "jrogers@unomaha.edu"
## [23] "aroslanowski@unomaha.edu"     "vrykov@unomaha.edu"
## [25] "gsand@unomaha.edu"            "larissaschroeder@unomaha.edu"
## [27] "aswift@unomaha.edu"           "kluhing@unomaha.edu"
## [29] "dvelcsov@unomaha.edu"         "ftorresvitor@unomaha.edu"
## [31] "congwang@unomaha.edu"         "ecook@unomaha.edu"
## [33] "cteller@unomaha.edu"          "sdowning@unomaha.edu"
## [35] "jheidel@unomaha.edu"          "maloney@cox.net"
## [37] "lstephens@unomaha.edu"        "zhenyuanwang@unomaha.edu"
## [39] "unomathematics@unomaha.edu"
```

d. Write a regular expression to extract words with 11 or more letters in the text. Include your codes and display the result that shows the words without duplication.

```
eleven_word <- unique(unlist(regmatches(myText,gregexpr("\\b[A-Za-z]+{11,}\\b",myTex
t))))
eleven_word
```

```
##   [1] "description"          "Mathematical"         "Statistical"
##   [4] "mathematics"          "webmanifest"          "msapplication"
##   [7] "googletagmanager"     "Intentionally"        "getElementsByTagName"
##  [10] "createElement"        "insertBefore"         "supplemental"
##  [13] "breadcrumbs"          "Communications"       "authorization"
##  [16] "SUPPLEMENTAL"         "autocomplete"         "placeholder"
##  [19] "Undergraduate"        "Mathematics"          "Opportunities"
##  [22] "opportunities"        "scholarships"         "Scholarships"
##  [25] "organizations"        "Organizations"        "internships"
##  [28] "Internships"          "publications"         "Publications"
##  [31] "MatheMavericks"       "conferences"          "Information"
##  [34] "registration"         "Registration"         "Accommodations"
##  [37] "nufoundation"         "transparent"          "Differential"
##  [40] "Quantitative"         "Visualization"        "Interactive"
##  [43] "Exploratory"          "jeffreydepue"         "Theoretical"
##  [46] "keithgallagher"       "Commutative"          "christopherjames"
##  [49] "christopher"          "Programming"          "michaelmatthews"
##  [52] "kenzimedeiros"        "Palayangoda"          "palayangoda"
##  [55] "lpalayangoda"         "Engineering"          "applications"
##  [58] "approximations"       "Roslanowski"          "roslanowski"
##  [61] "aroslanowski"         "Bioinformatics"       "Combinatorics"
##  [64] "Competition"          "larissaschroeder"     "Scholarship"
##  [67] "Probability"          "ftorresvitor"         "Optimization"
##  [70] "Multivariate"         "Distributions"        "Concentration"
##  [73] "zhenyuanwang"         "Nonadditive"          "Integration"
##  [76] "Announcements"        "assistantship"        "application"
##  [79] "Assistantship"        "joinhandshake"        "unomathematics"
##  [82] "currentColor"         "information"          "achievement"
##  [85] "development"          "Development"          "unobookstore"
##  [88] "communication"        "buffettinstitute"     "waterforfood"
##  [91] "ruralprosperityne"    "accessibility"        "Accessibility"
##  [94] "discriminate"         "orientation"          "affiliation"
##  [97] "retaliation"          "discrimination"       "nondiscrimination"
## [100] "scrollToTop"
```

e. Now we want to extract all the phone/fax numbers in `myText` . Write a regular expression that would do this. Demonstrate your codes showing the results without duplication.

```
phone_number <-unlist(regmatches(myText,gregexpr("\\b[+1]?[0-9]+{3}[\\.]+[0-9]+{3}
[\\.]+[0-9]+{4}\\b",myText)))
phone_number
```

```
##  [1] "402.554.4016" "402.554.2836" "402.554.2848" "402.554.7270" "402.554.2842"
##  [6] "402.554.3622" "402.554.6749" "402.554.2414" "402.554.4867" "402.554.3522"
## [11] "402.554.3431" "402.554.2831" "402.554.2734" "402.554.2839" "402.554.3558"
## [16] "402.554.6748" "402.554.2688" "402.554.7233" "402.554.2827" "402.554.3105"
## [21] "402.554.3117" "402.554.6745" "402.554.6746" "402.554.4012" "402.554.3295"
## [26] "402.554.2691" "402.554.4044" "402.554.3841" "402.554.2443" "402.554.3841"
## [31] "402.554.3430" "402.554.3430" "402.554.3522" "402.554.3522" "402.554.3841"
## [36] "402.554.3841" "402.554.2800" "402.554.2800"
```

f. The link of ggplot2 documentation is https://ggplot2-book.org/individual-geoms.html (https://ggplot2-book.org/individual-geoms.html) and we would like to get the list of individual ggplot2 geoms from there. Write a regular expression that would extract all the geoms names (geom_bar is one of them) from this link and display the unique geoms. How many unique geoms does the page list?

```
url <-"https://ggplot2-book.org/individual-geoms.html"
link <- readLines(url, warn = FALSE)
#tail(link)
geom_names<-unlist(regmatches(link,gregexpr("\\bgeom_[a-z]+\\b",link)))
geom_names
```

```
##  [1] "geom_area"    "geom_bar"     "geom_line"    "geom_line"    "geom_path"
##  [6] "geom_line"    "geom_path"    "geom_point"   "geom_point"   "geom_polygon"
## [11] "geom_rect"    "geom_tile"    "geom_raster"  "geom_rect"    "geom_tile"
## [16] "geom_raster"  "geom_tile"    "geom_text"    "geom_point"   "geom_text"
## [21] "geom_bar"     "geom_tile"    "geom_line"    "geom_area"    "geom_path"
## [26] "geom_polygon" "geom_path"    "geom_polygon" "geom_path"    "geom_line"
## [31] "geom_smooth"  "geom_boxplot" "geom_violin"
```

```
cat ("\n")
```

```
cat("Number of unique geoms the page contains is  :", length(unique(geom_names)))
```

```
## Number of unique geoms the page contains is  : 13
```

4. Big data problem: Download the sample of a big data `bigDataSample.csv` from Canvas. Note that the data is in csv format and compressed for easy handling. You may need to uncompress it before using. Now answer the following questions.

a. Read the data and select only the columns that contains the word ‘human’. Store the data in an object `dat` . Report first few rows of your data.

```
sample_dat <-read.csv("bigDataSample.csv")
# Select columns containing the word 'human'
dat <- sample_dat %>%
  dplyr::select(contains("human"))
# Report the first few rows of the selected columns
head(dat)
```

```
##   var_human_1_g var_human_1_p var_human_1_b var_human_1_e var_human_1_n
## 1      18.99545            21             1    21.6321136      26.03268
## 2      15.02303            34             3     0.3838458      26.92529
## 3      37.44410            28             2    33.4801022      39.30039
## 4      36.33714            26             2     2.8761174      33.75177
## 5      21.06330            25             1     3.1657313      26.19248
## 6      16.52637            35             2     5.3108922      25.07192
```

b. The data frame `dat` should have 5 columns. Rename the column names keeping only the last character of the column names. So each column name will have only one character. Report first few rows of your data now.

```
dat <- dat %>%
  dplyr::rename_all(~ substr(., nchar(.), nchar(.)))
head(dat)
```

```
##          g  p b          e        n
## 1 18.99545 21 1 21.6321136 26.03268
## 2 15.02303 34 3  0.3838458 26.92529
## 3 37.44410 28 2 33.4801022 39.30039
## 4 36.33714 26 2  2.8761174 33.75177
## 5 21.06330 25 1  3.1657313 26.19248
## 6 16.52637 35 2  5.3108922 25.07192
```

c. Compute and report the means of each columns group by column b in a nice table.

```
library(dplyr)
mean_by_b <- dat %>%
   group_by(b)%>%
   summarize(across(everything(), mean, na.rm = TRUE))
mean_by_b <-round(mean_by_b ,2)
library(DT)
datatable(data=,mean_by_b)
```

Show 10 entries Search:

	b	g	p	e	n
1	0	28.75	23.76	12.21	29.44
2	1	22.48	25.28	10.42	29.34
3	2	23.85	24.95	9.62	30.63
4	3	23.81	25.41	10.48	30.25

Showing 1 to 4 of 4 entries Previous 1 Next

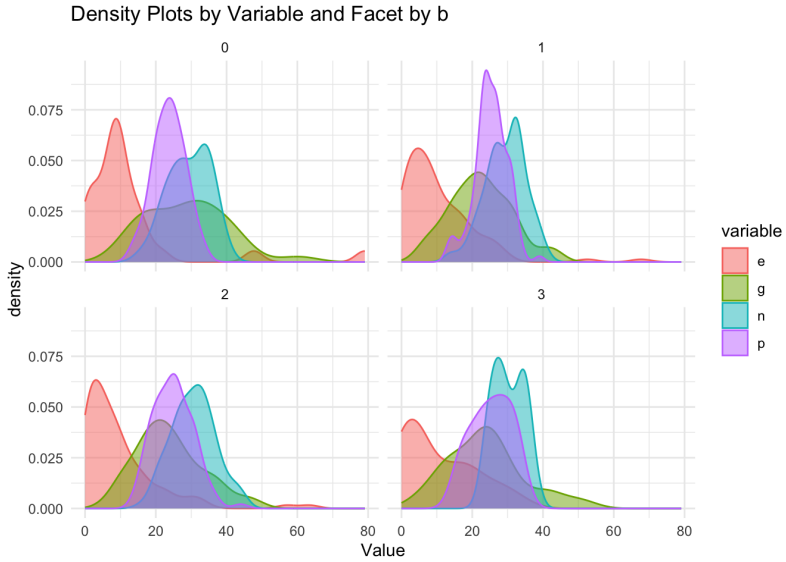
d. Change the data into long form using id='b' and store the data in `mdat` . Report first few rows of data.

```
library(tidyr)
mdat <- dat %>%
  pivot_longer(cols = -b, names_to = "variable", values_to = "value")
head(mdat,5)
```

```
## # A tibble: 5 × 3
##       b variable value
##   <int> <chr>    <dbl>
## 1     1 g         19.0
## 2     1 p         21
## 3     1 e         21.6
## 4     1 n         26.0
## 5     3 g         15.0
```

e. The data frame `mdat` is now ready for plotting. Generate density plots of value, color and fill by variable and facet by b.

```
library(ggplot2)
#Generate density plots
ggplot(mdat, aes(x = value, fill = variable, color = variable)) +
  geom_density(alpha = 0.5) +
  facet_wrap(~ b) +
  labs(title = "Density Plots by Variable and Facet by b", x = "Value") +
  theme_minimal()
```



f) This data set is a sample of much bigger data set. Here we read the data set and then selected the desired column. Do you think it would be wise do the same thing with the actual larger data set? Explain how you will solve this problem of selecting few columns (as we did in question a) without reading the whole data set first. Demonstrate that showing your codes.

i will use `fread` ,as `fread` memory maps the file into memory and then iterates through the file using pointers.Whereas `read.csv` reads the file into a buffer via a connection. So `fread` is faster.

```
library(data.table)
dat_new <- fread("bigDataSample.csv", select = c("var_human_1_g","var_human_1_p","var
_human_1_b","var_human_1_e","var_human_1_n"))
head(dat_new,5)
```

```
##    var_human_1_g var_human_1_p var_human_1_b var_human_1_e var_human_1_n
## 1:      18.99545            21             1    21.6321136      26.03268
## 2:      15.02303            34             3     0.3838458      26.92529
## 3:      37.44410            28             2    33.4801022      39.30039
## 4:      36.33714            26             2     2.8761174      33.75177
## 5:      21.06330            25             1     3.1657313      26.19248
```